EC4206: SOFTWARE TESTING AND

QUALITY ASSURACE

FINAL PROJECT

:

:

:

:

NAME

REG NO

GROUP NO

DATE

KODITHUWAKKU P.K.E.

EG/2022/5148

CE- 15

05/08/2025

# Test-Driven Development (TDD) & Behavior-Driven Development (BDD)

## TDD

### **Identify at least two core features (e.g., add task, validate user input).**

Chosen Features:

* Save Course – Admin adds a new course.
* Update Course – Update an existing course.

#### SAVE COURSE

##### **Write unit tests first using JUnit (Spring Boot) or NUnit (.NET) before implementing the features.**

Create file-> src/test/java/com/brightpath/backend/service/CourseServiceTest.java

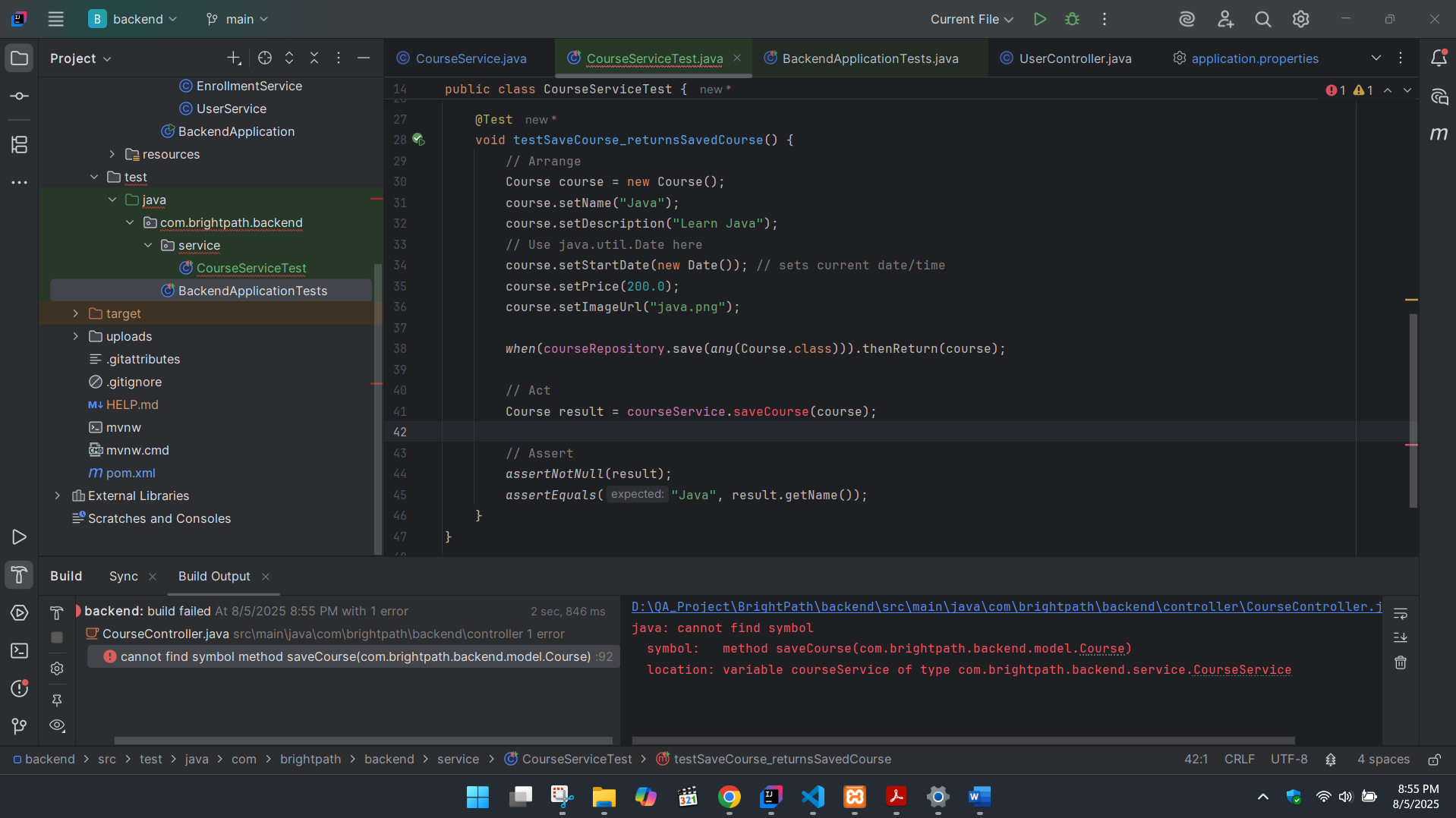
CourseServiceTest.java

package com.brightpath.backend.service;  
  
import com.brightpath.backend.model.Course;  
import com.brightpath.backend.repository.CourseRepository;  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
import org.mockito.\*;  
  
import java.util.Date;  
  
import static org.junit.jupiter.api.Assertions.\*;  
import static org.mockito.Mockito.\*;  
  
public class CourseServiceTest {  
  
 @Mock  
 private CourseRepository courseRepository;  
  
 @InjectMocks  
 private CourseService courseService;  
  
 @BeforeEach  
 void setUp() {  
 MockitoAnnotations.*openMocks*(this);  
 }  
  
 @Test  
 void testSaveCourse\_returnsSavedCourse() {  
 // Arrange  
 Course course = new Course();  
 course.setName("Java");  
 course.setDescription("Learn Java");  
 // Use java.util.Date here  
 course.setStartDate(new Date()); // sets current date/time  
 course.setPrice(200.0);  
 course.setImageUrl("java.png");  
  
 *when*(courseRepository.save(*any*(Course.class))).thenReturn(course);  
  
 // Act  
 Course result = courseService.saveCourse(course);  
  
 // Assert  
 *assertNotNull*(result);  
 *assertEquals*("Java", result.getName());  
 }  
}

##### **Follow the Red-Green-Refactor cycle:**

* Write a failing test (Red).

Write the test first, but the saveCourse() method does **not exist** in CourseService.

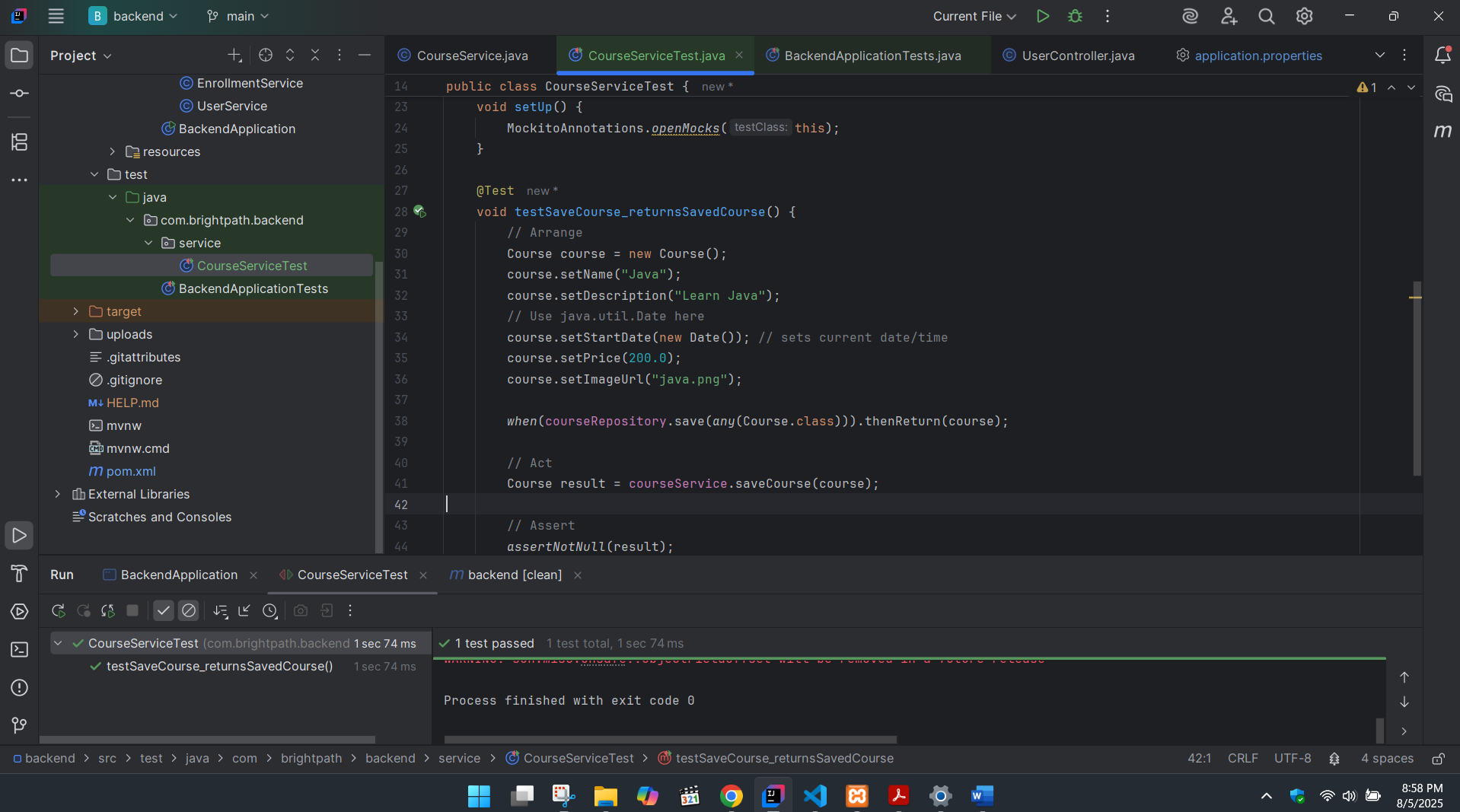


* Implement minimum code to pass the test (Green).

public Course saveCourse(Course course) {

return courseRepository.save(course);

}



* Refactor the code to improve quality while keeping tests green

Add this to CourseService.java

public Course saveCourse(Course course) {

if (course.getName() == null || course.getName().isBlank()) {

throw new IllegalArgumentException("Course name is required");

}

return courseRepository.save(course);

}

And also add this to CourseServiceTest.java

@Test

void testSaveCourse\_withEmptyName\_shouldThrowException() {

Course course = new Course();

course.setName(""); // Invalid name

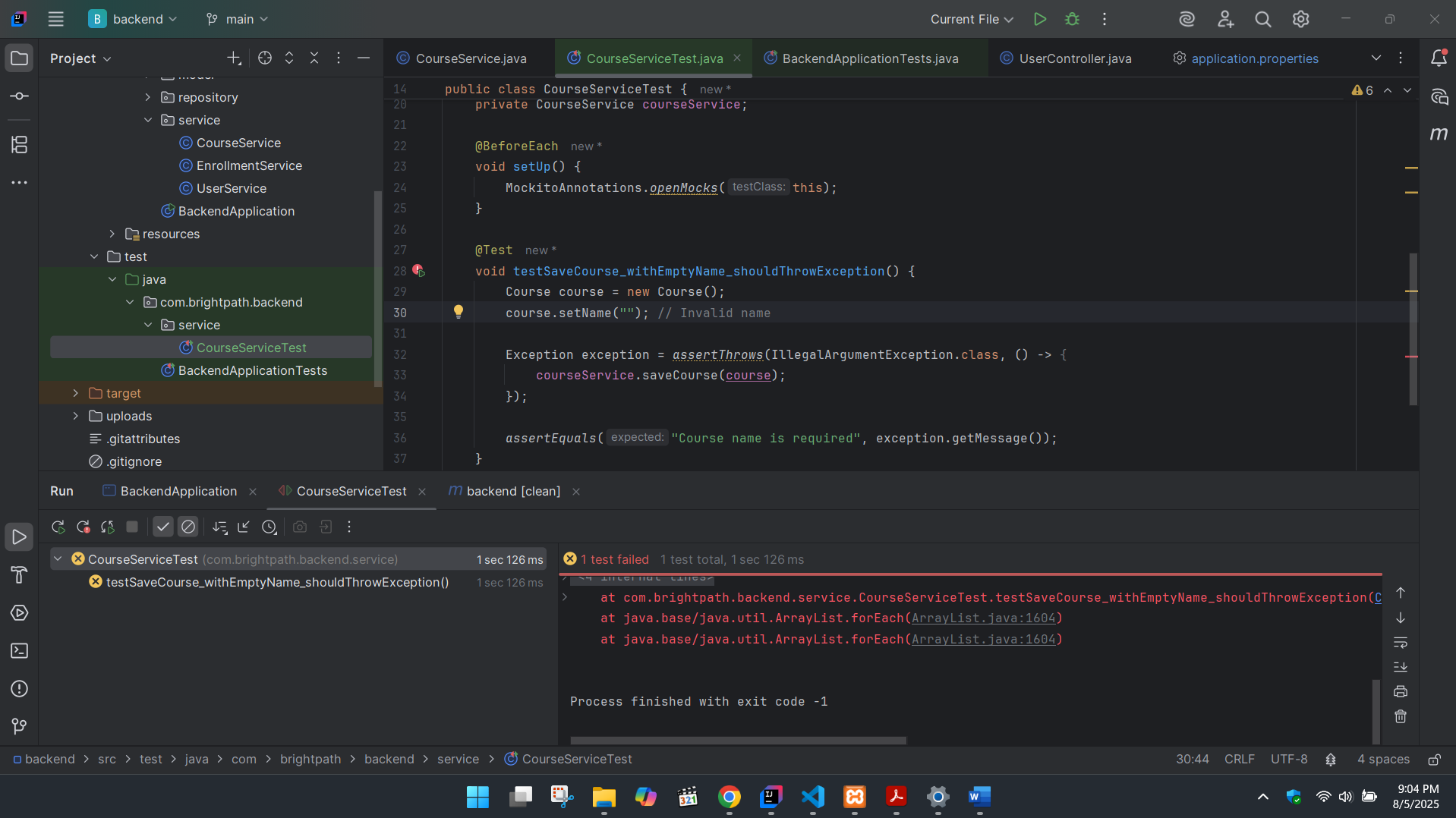
Exception exception = assertThrows(IllegalArgumentException.class, () -> {

courseService.saveCourse(course);

});

assertEquals("Course name is required", exception.getMessage());

}



##### **Explain and demonstrate the steps.**

**TDD: Red-Green-Refactor Process for saveCourse() Method**

| **Step** | **Description** |
| --- | --- |
| **Red** | I first wrote a unit test for the saveCourse() method, but the method did not exist yet. This caused a compilation error or test failure — which is expected in TDD. |
| **Green** | I then implemented just enough code to pass the test, by calling courseRepository.save(course). The test passed successfully. |
| **Refactor** | I improved the method by adding input validation for course name. I also added an extra unit test to check for invalid input. All tests still passed after refactoring. |

#### UPDATE COURSE

##### 

@Test

void testUpdateCourse\_shouldReturnUpdatedCourse() {

// Arrange

Long courseId = 1L;

Course existingCourse = new Course();

existingCourse.setId(courseId);

existingCourse.setName("Old Name");

existingCourse.setDescription("Old Desc");

// Convert LocalDate to java.util.Date

LocalDate oldDate = LocalDate.of(2025, 8, 1);

LocalDate newDate = LocalDate.of(2025, 8, 15);

ZoneId zone = ZoneId.systemDefault();

existingCourse.setStartDate(Date.from(oldDate.atStartOfDay(zone).toInstant()));

existingCourse.setPrice(100.0);

existingCourse.setImageUrl("old.png");

Course updatedDetails = new Course();

updatedDetails.setName("New Name");

updatedDetails.setDescription("New Desc");

updatedDetails.setStartDate(Date.from(newDate.atStartOfDay(zone).toInstant()));

updatedDetails.setPrice(200.0);

updatedDetails.setImageUrl("new.png");

when(courseRepository.findById(courseId)).thenReturn(Optional.of(existingCourse));

when(courseRepository.save(any(Course.class))).thenReturn(existingCourse);

// Act

Course result = courseService.updateCourse(courseId, updatedDetails);

// Assert

assertNotNull(result);

assertEquals("New Name", result.getName());

assertEquals("New Desc", result.getDescription());

assertEquals(Date.from(newDate.atStartOfDay(zone).toInstant()), result.getStartDate());

assertEquals(200.0, result.getPrice());

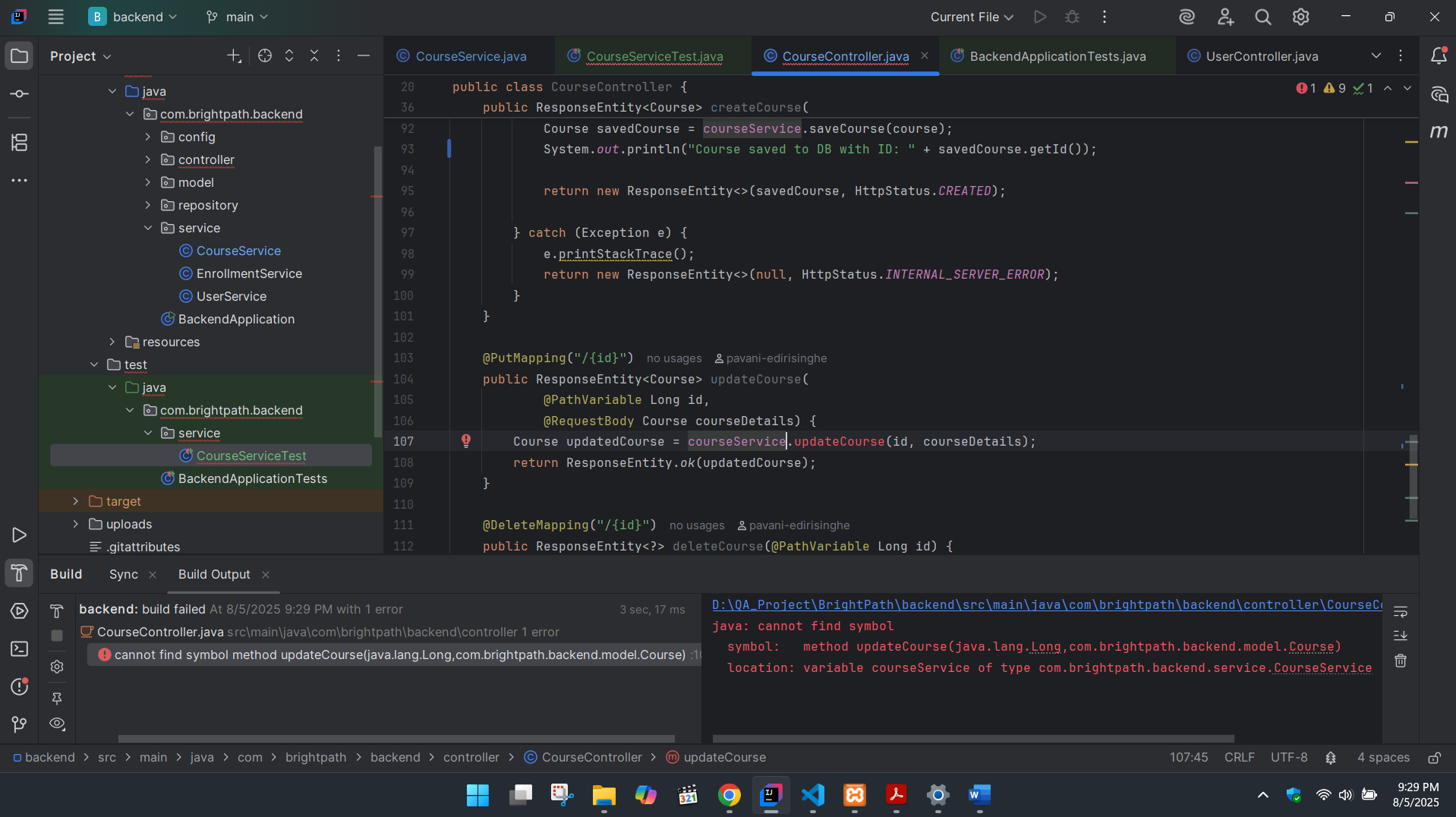
assertEquals("new.png", result.getImageUrl());

}

##### **Follow the Red-Green-Refactor cycle:**

* Write a failing test (Red).

Write the test first, but the updateCourse() method does **not exist** in CourseService.



* Implement minimum code to pass the test (Green).

public Course updateCourse(Long id, Course courseDetails) {

Course course = courseRepository.findById(id)

.orElseThrow(() -> new RuntimeException("Course not found"));

course.setName(courseDetails.getName());

course.setDescription(courseDetails.getDescription());

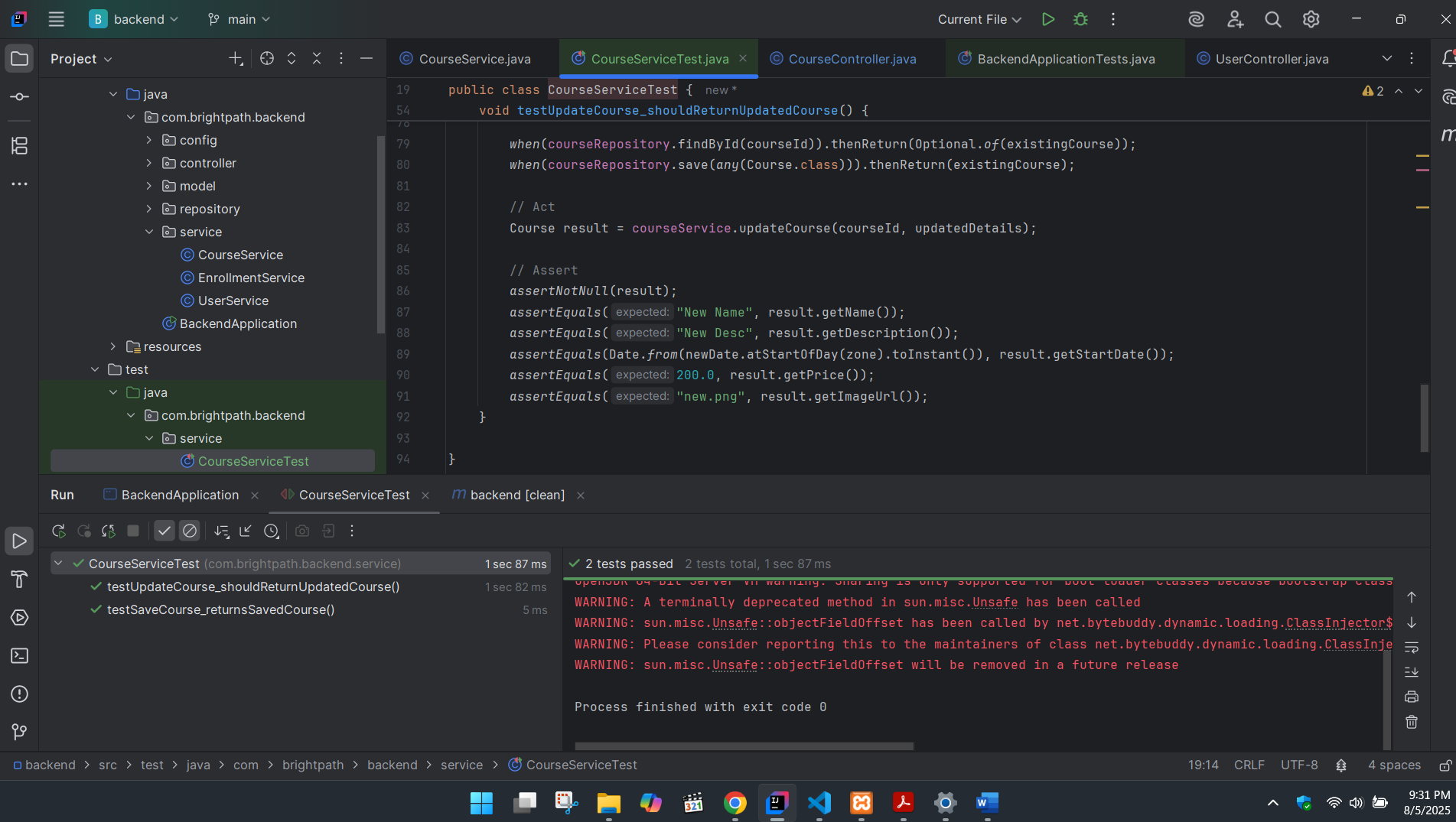
course.setStartDate(courseDetails.getStartDate());

course.setPrice(courseDetails.getPrice());

course.setImageUrl(courseDetails.getImageUrl());

return courseRepository.save(course);

}



* Refactor the code to improve quality while keeping tests green

Add this to CourseService.java

public Course updateCourse(Long id, Course courseDetails) {

Course course = courseRepository.findById(id)

.orElseThrow(() -> new RuntimeException("Course not found with ID: " + id));

if (courseDetails.getName() == null || courseDetails.getName().isBlank()) {

throw new IllegalArgumentException("Course name cannot be blank");

}

course.setName(courseDetails.getName());

course.setDescription(courseDetails.getDescription());

course.setStartDate(courseDetails.getStartDate());

course.setPrice(courseDetails.getPrice());

course.setImageUrl(courseDetails.getImageUrl());

return courseRepository.save(course);

}

And also add this to CourseServiceTest.java

@Test

void testUpdateCourse\_withBlankName\_shouldThrowException() {

Long courseId = 1L;

Course existingCourse = new Course();

existingCourse.setId(courseId);

existingCourse.setName("Old Name");

Course invalidDetails = new Course();

invalidDetails.setName(""); // Invalid name

when(courseRepository.findById(courseId)).thenReturn(Optional.of(existingCourse));

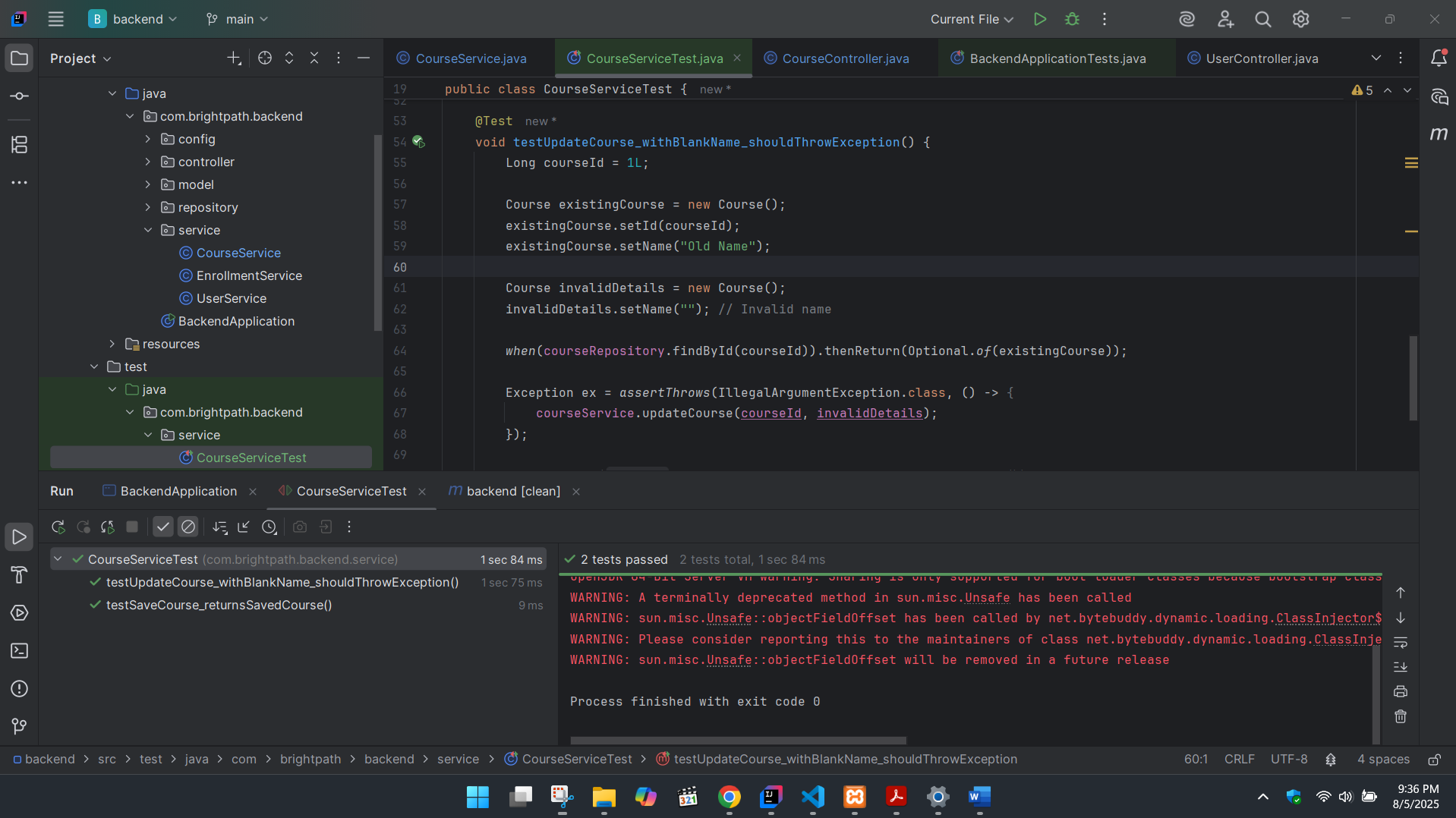
Exception ex = assertThrows(IllegalArgumentException.class, () -> {

courseService.updateCourse(courseId, invalidDetails);

});

assertEquals("Course name cannot be blank", ex.getMessage());

}



##### **Explain and demonstrate the steps.**

### Explain Red-Green-Refactor for updateCourse()

| **Phase** | **Description** |
| --- | --- |
| Red | Wrote a test for updateCourse() before implementing it. It failed because the method didn't exist. |
| Green | Implemented just enough code to pass the test — updated the course and saved it. |
| Refactor | Added validation for blank names and wrote additional test cases to improve robustness. |

## BDD

## 

### **Write feature files in Gherkin syntax describing at least one user story**

Create src/test/resources/features/add\_course.feature

Feature: Add Course  
  
 Scenario: Successfully adding a new course  
 Given the course name is "ReactJS"  
 And the description is "Learn frontend development"  
 And the start date is "2025-08-20"  
 And the price is 250.0  
 And the image URL is "react.png"  
 When I submit the course  
 Then the course should be saved successfully  
 And the course name should be "ReactJS"

### 

### **Implement step definitions in Java with Cucumber or .NET BDD framework.**

src/test/java/com/brightpath/backend/bdd/AddCourseSteps.java

package com.brightpath.backend.bdd;  
  
import com.brightpath.backend.model.Course;  
import com.brightpath.backend.repository.CourseRepository;  
import com.brightpath.backend.service.CourseService;  
import io.cucumber.java.Before;  
import io.cucumber.java.en.\*;  
import org.mockito.\*;  
  
import java.time.LocalDate;  
import java.time.ZoneId;  
import java.util.Date;  
  
import static org.junit.jupiter.api.Assertions.\*;  
import static org.mockito.Mockito.\*;  
  
public class AddCourseSteps {  
  
 private Course course; // the course being created  
 private Course savedCourse; // the course after saving  
  
 @Mock  
 private CourseRepository courseRepository;  
  
 @InjectMocks  
 private CourseService courseService;  
  
 @Before // Use Cucumber's @Before, not JUnit's @BeforeEach  
 public void setup() {  
 MockitoAnnotations.*openMocks*(this);  
 course = new Course(); // Properly initialize  
 }  
  
 @Given("the course name is {string}")  
 public void the\_course\_name\_is(String name) {  
 course.setName(name);  
 }  
  
 @And("the description is {string}")  
 public void the\_description\_is(String desc) {  
 course.setDescription(desc);  
 }  
  
 @And("the start date is {string}")  
 public void the\_start\_date\_is(String dateStr) {  
 LocalDate localDate = LocalDate.*parse*(dateStr);  
 Date date = Date.*from*(localDate.atStartOfDay(ZoneId.*systemDefault*()).toInstant());  
 course.setStartDate(date);  
 }  
  
 @And("the price is {double}")  
 public void the\_price\_is(Double price) {  
 course.setPrice(price);  
 }  
  
 @And("the image URL is {string}")  
 public void the\_image\_url\_is(String url) {  
 course.setImageUrl(url);  
 }  
  
 @When("I submit the course")  
 public void i\_submit\_the\_course() {  
 *when*(courseRepository.save(*any*(Course.class))).thenReturn(course);  
 savedCourse = courseService.saveCourse(course);  
 }  
  
 @Then("the course should be saved successfully")  
 public void the\_course\_should\_be\_saved\_successfully() {  
 *assertNotNull*(savedCourse);  
 }  
  
 @And("the course name should be {string}")  
 public void the\_course\_name\_should\_be(String expectedName) {  
 *assertEquals*(expectedName, savedCourse.getName());  
 }  
}

### **Automate the scenarios and run the BDD tests.**

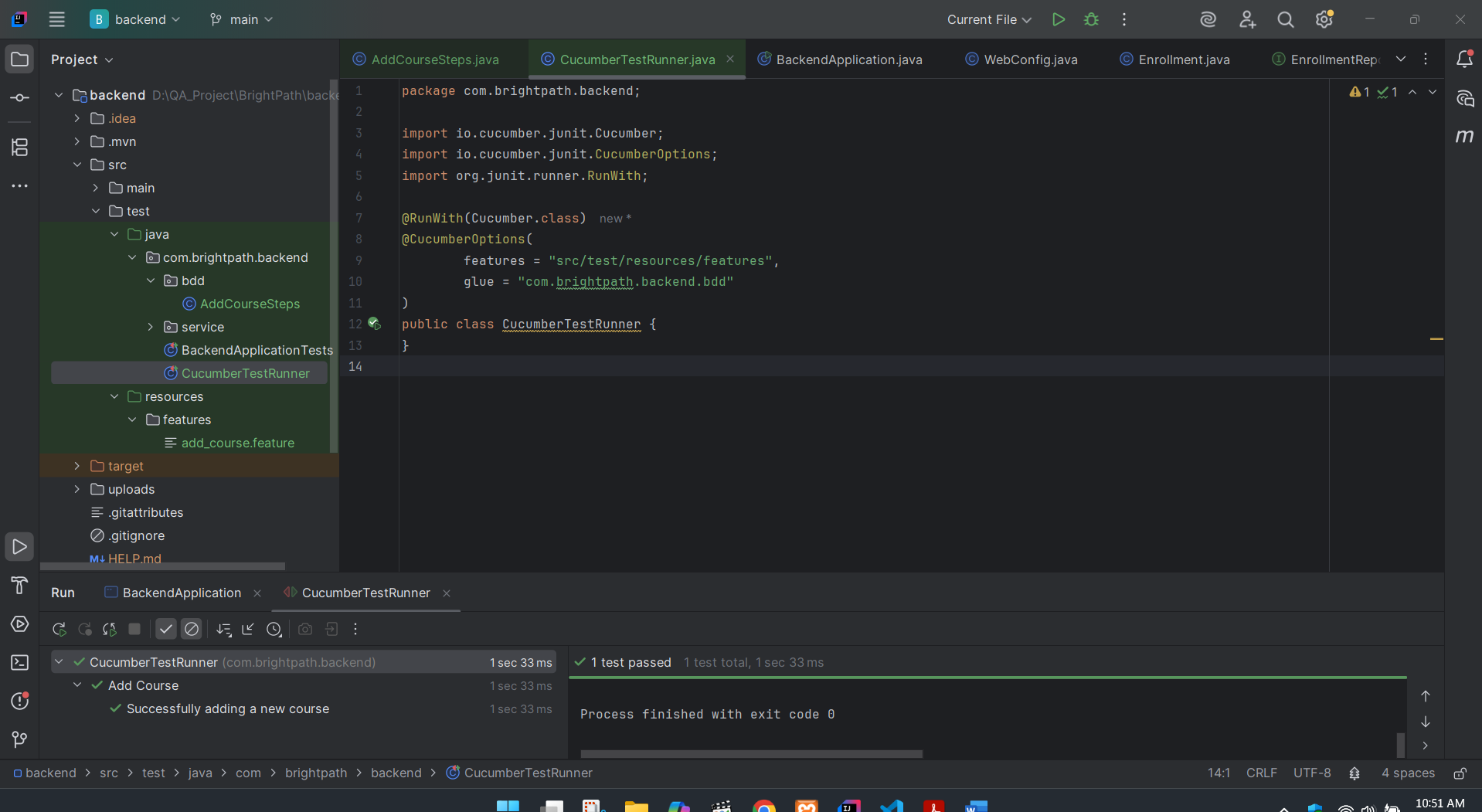
src/test/java/com/brightpath/backend/CucumberTestRunner.java

package com.brightpath.backend;  
  
import io.cucumber.junit.Cucumber;  
import io.cucumber.junit.CucumberOptions;  
import org.junit.runner.RunWith;  
  
@RunWith(Cucumber.class)  
@CucumberOptions(  
 features = "src/test/resources/features",  
 glue = "com.brightpath.backend.bdd"  
)  
public class CucumberTestRunner {  
}

### **Demonstrate the test run and results.**

* Gherkin feature file (add\_course.feature)
* Step definitions in Java
* IntelliJ green test output after running CucumberTestRunner
* Final result: Cucumber passes the test scenario

Run the CucumberTestRunner



# Test Automation & Continuous Integration

## 

## Write the following automated tests:

### **2 Selenium UI test scripts:**